

Modeling and Analysis of Procedural Security

Komminist Sisai Weldemariam
Fondazione Bruno Kessler and University of Trento
Via Sommarive 14
38100 Trento (TN), Italy
sisai@fbk.eu

ABSTRACT

This paper describes the ongoing work for tackling security at procedural/organizational level. Namely, we are trying to understand weaknesses and strengths of the procedures regulating systems and system processes, in order to analyze possible attacks and their effects. We are carrying out this activity by trying providing formal specifications of the procedures and using model checkers to help us derive possible attacks. To do so, we present a methodology and techniques we are devising and experimenting with to tackle problem highlighted above. We believe such an analysis to be essential to, first, identify the security boundaries, that is, the conditions under which procedures can be carried out securely and, secondly, to devise a set of requirement to be applied both at the organizational level and on the (software) systems used.

1. INTRODUCTION

Security is a complex non-functional requirement whose implementation requires intervention over the entire parts of a system at all levels of detail. As our society is getting more dependent on system, security is becoming one of the important constraints. Security threats are those that breach or violate the security goals to compromise or invalidate the confidentiality, integrity, and availability of the systems and system processes. It is widely reported that (see, for instance, in [1, 2, 3]), all systems and system processes making use of information and communication technologies (ICT) are exposed to security threats in one way or another.

Procedures in a system often define how critical assets are to be managed, elaborated, and transformed in a meaningful way. They are devised to ensure that systems or system processes are carried out correctly and securely. Equally, incorrect or malicious “deviations” from the procedures defined by the law may result in violations of fundamental security and privacy features of the system. In electoral systems, for instance, these deviations can cause severe problems like the violation of the rights of citizens (e.g. secrecy of vote) or,

even, in threats to the integrity of electoral data and of the election results. Therefore, in order to consider and analyze what happens when a procedure is not followed as prescribed and has to define mechanisms to ensure that violations can be detected, procedures should be considered as important as the technical security features of the systems.

We are approaching the problem by reasoning about the procedures and controls that regulate the usage of system or system processes. We focus on techniques and tools to analyze security threats at the organizational and procedural level in order to highlight possible “(un)detectable” threats that can be realistically carried out. The strategies we follow are by trying finding intersection among business process modeling, security and formal method techniques. Namely, by providing formal models of the procedures, by “injecting” threats in such models and by analyzing, through the help of model checkers, the effects of such threats.

2. PROCEDURAL SECURITY ANALYSIS

Procedural security deals with the identification, modeling, establishment, and enforcement of security policies about the procedures that regulate the usage of a system and system processes. The breach of security objectives during the execution of the procedures is known as *threat to the procedures* (or *procedural threats*). We call *procedural security analysis* the process of understanding the impact and effects of procedural threats, namely courses of actions that can take place during the execution of the procedures, and which are meant to alter, in an unlawful way, the assets manipulated by procedures.

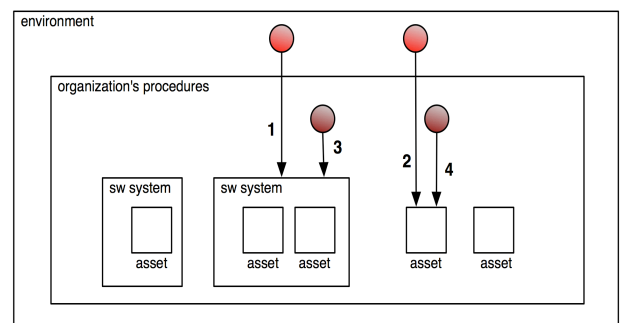


Figure 1: Procedural Security Analysis

The situation is depicted in Figure 1. Our *target of evaluation* is a (complex) organizational setting in which procedures transform and elaborate *assets*, which may not necessarily be just digital (e.g. a printed ballot). The procedures are meant to add value to the assets and to protect them from attacks, which can either come from external sources or from insiders. In particular, we distinguish the following kinds of attacks:

1. *Attacks on digital assets* (item 1 and item 3 in Figure 1). These kinds of attacks are meant to alter one or more of the digital assets of an organization. Attacks can either be carried out from external sources (the environment) or from internal sources. Opportunities for attacks are determined by the organizational setting and by the security provided by the digital systems.
2. *Attacks on other kind of assets* (item 2 and item 4 in Figure 1). These attacks are meant to alter one or more of the non-digital assets of an organization. Attacks can either be carried out from external sources (the environment) or from internal sources or a combination of both that it forms coordinated attacks. Opportunities for attacks are determined by the organizational settings only.

Security assessment (like [3, 4]) usually focuses on understanding items 1 and 3, namely, types and effects of attacks on (software) systems. In the next section we propose a tool-supported methodology to tackle also points 2 and 4 above, namely types and effects of attacks on assets that are not (necessarily) digital and that derive from the way in which procedures are implemented and carried out.

3. THE METHODOLOGY

In this section, we describe the methodology we devise to perform procedural security analysis (See Figure 2 also).

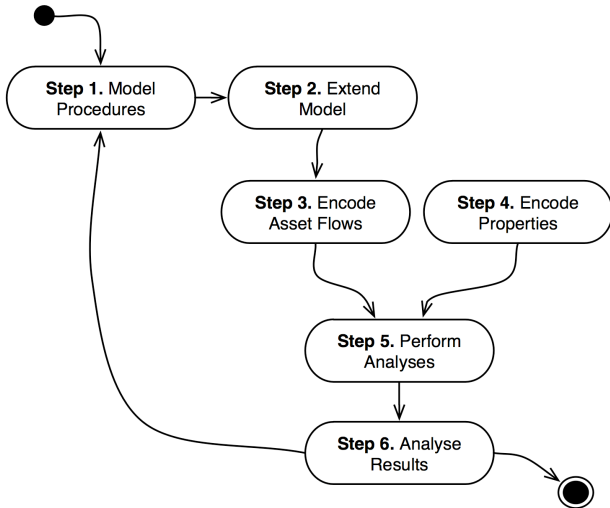


Figure 2: The process of formal procedural security.

1. The first step consists of *providing (business) models of the procedures* under evaluation (Step 1 of Figure 2). These models are meant to describe the process or the processes to be analyzed, and to define how assets are elaborated and transformed by such procedures. In order to ease the task of translating the models into executable specification, we decided to stick to a subset of the UML, that allows us to describe the domain concepts in a strict and defined way (see [5]). The model contains information about the procedures (workflows), the assets used, their features, and the actors and their role when participating to different workflows.
2. The second step consists of *“injecting“ threat actions* into the model and generate what we call *extended model* (Step 2 of Figure 2). A threat is an action that alters some features of an assets or allows some actors privileges (e.g. a “read” privilege) on some assets. Thus, in the extended model, not only assets are modified according to what the procedures define, but they can also be transformed by the (random) execution of one or more threat actions. Since attacks depend on what threat-actions are carried out, the effectiveness of the analysis depends upon the threat actions that are defined and the injection strategy that is chosen. With respect to the first point (threat action), we allow attackers any possible privilege and action to assets. With respect to the second point, it turns out that the best and most general strategy is that of injecting all possible threat-actions at all possible steps of the nominal procedures (the model checker will then take care of “pruning” useless threats, namely threats which do not lead to any successful attack). The construction of the extended model, whose generation can be automated, is currently performed by hand.
3. The third step is *encoding the asset model into executable specification* (Step 3 of Figure 2). From the extended models defined at the previous step we derive executable specification that define how each asset is manipulated by the procedures. An asset flow encodes how the features of an asset get changed by the execution of workflows. Asset flows are represented in the NuSMV input language [6]. The NuSMV model of the asset flows is based on the definition of “program counters” that ensure that procedures are executed according to the specifications, and by defining one module per asset with one state variable per asset-feature. The state variables encode how features change during the execution of the procedures. Accessory information, such as actors responsible for the different activities can be used, e.g., to express security properties, which, in turn, are verified against the asset model. The necessity of modeling actors roles in NuSMV depends upon the target of the security analysis.
4. The fourth step is *specifying security properties to model check*. The specification of the desired (procedural) security properties, namely, the security goals that have to be satisfied are then encoded using LTL/CTL formula. Asset flows and security properties are then the input to NuSMV.

5. In step five (Step 5 of Figure 2) we *perform analyses*, namely, we run the model checker and collect results (counter-examples). Counter-examples of security properties found by the model checker encode sequences of actions that, if executed, pose a threat to security of one or more assets. In standard situations, the counter-example will contain the execution of one (or more) asset threat. A counter-example in which no asset threats need to be executed would show an inherent weakness in the *nominal* workflows or, otherwise, an error in the specifications.
6. The last step consists in *analyzing the obtained results* (Step 6 of Figure 2), by looking at the counter-examples generated by the model-checker. In particular some of the information we are interested in is:
 - (a) The **Actor-Play-Role** namely, the roles actors have in the execution of the attack and the privileges they get on assets.
 - (b) **Reachability** the sequence of actions leading to the violation of a security goal, with particular respect to the execution of asset-threats.

This step allows us to achieve two goals. First, it allows us to understand what are the hypotheses and conditions under which a given security goal is achieved or breached. Second, it provides information to try and modify the existing procedures, so that security breaches can be taken care of.

The analysis approach we take is very similar to that of FSAP/NuSMV-SA [7], for safety analysis, whereby a system specification is “enriched” with information about faults and analyzes are carried out to understand the effect and impact of faults on safety requirements expressed in the form of LTL/CTL formulae. Analogously to what happens in safety analysis when analyzing, e.g., the loss of a critical functions, enhancing the procedures results in reducing the probability of an attack or making the attack more complex, rather than eliminating it.

4. RELATED WORK

Various approaches for specifying, modeling, analyzing, and assessing security have been proposed in the past and have been proven useful for zeroing the security lacks of the software systems under analysis (see, for instance, [3, 8, 4, 9]). However, these techniques are less or otherwise not effective in what we call *procedurally rich* scenarios, namely situations in which software systems are just part of a more complex organizational setting, in which procedures have to be executed on security-critical assets (belonging both to the digital and to the physical realm). This is exactly the case of (voting and) e-voting, in which even in those countries that have adopted a high level of automation, the executions of procedures and controls, carried out by people on physical assets (e.g. printouts of the digital votes), remains a necessary and unavoidable part.

We discuss here how our work relates with the CORAS methodology, with attack trees, and attack graphs.

The CORAS methodology (see, for instance, [10, 4, 11, 12]) is based on a profile of the UML, that allows to represent graphically the system under analysis and threat scenarios. The methodology defines also a process to elicit information from stakeholders and a tool, to visually represent the analysis assessment. The focus of CORAS, however, is more on representational and on elicitation aspects, rather than on automation of the security analysis process.

Attack trees are a well-structured methodology and notation for analyzing the security of systems, subsystems, and system processes in a way which is analogous to fault trees in the safety domain. Many variants of attack trees have been proposed — for instance, Barynov and Jadhwal [13] propose a formal model of attack trees that allows to represent and analyze coordinated attacks in terms of multiple actions; Fovino and Masera [3] introduce a multidimensional view of attack trees that allows to attach context information to the boundary knowledge on each attack tree. The use of goal-orientation for modeling and analysis of security threats is described in [14] —notably, the notion of “goal” to represent the sequence of an adversary actions and subsequently reason on the possibilities of goal satisfaction is well described. Differently from ours, the approaches discussed above do not consider security threats caused by procedural weaknesses in their modeling and analysis process. Moreover, our approach allows for a simpler integration of notions such as threats caused by interactions of roles, co-operations among actors, and multi-step attacks. These kind of attacks turn out to be extremely relevant in the applicative domain we are interested in.

The application of model checking to automatically generate and analyze attack graphs is presented in [15] — in particular, by representing network systems under analysis as state machines and security objectives as properties to check with a model checker. In this context attack graphs are generated as counterexamples. Our approach complements and extends it, by lifting the context to that of procedures and by focusing on asset-flows.

From the application standpoint, various work have been going on the representation of e-voting procedures with business process notations. In this area, the work closest in spirit to ours is [16, 17], where the authors argue the need for procedural security in electronic elections and provide various examples of procedural risks occurred during trials in the UK. The same authors in [18] also investigate the need for applying business process re-engineering to electoral process. Our focus, however, is on the technical machinery to automate the analysis.

5. CONCLUSION AND FUTURE WORK

The work presented in this paper is a part of a doctoral program concerned with the modeling and analyzing of procedural security in complex systems. To that end, we mentioned (e)Voting systems as an applicative domain where procedural security is more effective.

We describe a methodology to perform procedural security analysis based on explicit reasoning on asset flows —notably, by building a model to describe the nominal procedures under analysis and, injecting possible threat actions (by as-

suming that any combination of threats can be possible in all steps) into the model.

Among the advantages, a structured approach to analyze security of procedures and a general threat injection strategy that allows a high level of generality in defining the attacks. The usage of model checkers, moreover, allows for reasoning about threat composition and to highlight, e.g. the level of coordination and resources required to carry out certain attacks.

There are limitations exist to the current approach. First, the approach lacks automation. Second, the approach does not explicitly describe how to represent and model threats. Third, bigger scale experimentation is needed to evaluate and show the significance of the proposed approach. In fact, we experimented the approach through a simple snippet example, which is derived from the procedure in place for e-voting trials.

However, the work presented in this paper is ongoing. Among the areas of development we mention automation (e.g. algorithms to automatically perform threat injection) and better tool support.

6. REFERENCES

- [1] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Carl Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Transactions on Dependable and Secure Computing*, 01(1):11–33, 2004.
- [2] S. Myagmar, A. Lee, and W. Yurcik. Threat Modeling as a Basis for Security Requirements. In *StorageSS '05: Proceedings of the 2005 ACM workshop on Storage security and survivability*, pages 94–102, New York, NY, USA, 2005. ACM Press.
- [3] Igor Nai Fovino and Marcelo Masera. Through the Description of Attacks: A Multidimensional View. In Janusz Górski, editor, *SAFECOMP*, volume 4166 of *Lecture Notes in Computer Science*, pages 15–28. Springer, 2006.
- [4] Monika Vetterling, Guido Wimmel, and Alexander Wisspeintner. A Graphical Approach to Risk Identification, Motivated by Empirical Investigations. *Lecture Notes in Computer Science*, pages 574–588, Thursday, November 23 2006.
- [5] Komminist Weldemariam, Adolfo Villafiorita, and Andrea Mattioli. Assessing Procedural Risks and Threats in e-Voting: Challenges and an Approach. In Ammar Alkassar and Melanie Volkamer, editors, *VOTE-ID*, volume 4896 of *Lecture Notes in Computer Science*, pages 38–49. Springer, 2007.
- [6] P. Bertoli, A. Cimatti, M. Pistore, M. Roveri, and P. Traverso. NuSMV 2: An Open Source Tool for Symbolic Model Checking. In *Proc. of International Conference on Computer-Aided Verification*, 2002.
- [7] Marco Bozzano and Adolfo Villafiorita. The FSAP/NuSMV-SA Safety Analysis Platform. *Int. J. Softw. Tools Technol. Transf.*, 9(1):5–24, 2007.
- [8] David Basin, Jürgen Doser, and Torsten Lodderstedt. Model Driven Security for Process-Oriented Systems. In *SACMAT '03: Proceedings of the eighth ACM symposium on Access control models and technologies*, pages 100–109, New York, NY, USA, 2003. ACM.
- [9] Guido Oliver Wimmel. *Model-Based Development of Security-Critical Systems*. PhD thesis, Technische Universität München, June 2005.
- [10] Rune Fredriksen, Monica Kristiansen, Bjørn Axel Gran, Ketil Stølen, Tom Arthur Opperud, and Theodosios Dimitrakos. The CORAS Framework for a Model-Based Risk Management Process. In *SAFECOMP '02: Proceedings of the 21st International Conference on Computer Safety, Reliability and Security*, pages 94–105, London, UK, 2002. Springer-Verlag.
- [11] Theodosios Dimitrakos, Brian Ritchie, Dimitris Raptis, Jan Øyvind Agedal, Folker den Braber, Ketil Stølen, and Siv Hilde Houmb. Integrating Model-based Security Risk Management into eBusiness Systems Development: The CORAS Approach. In *I3E '02: Proceedings of the IFIP Conference on Towards The Knowledge Society*, pages 159–175, Deventer, The Netherlands, The Netherlands, 2002. Kluwer, B.V.
- [12] Fredrik Vraalsen, Mass Soldal Lund, Tobias Mahler, Xavier Parent, and Ketil Stølen. Specifying Legal Risk Scenarios Using the CORAS Threat Modelling Language. In Peter Herrmann, Valérie Issarny, and Simon Shiu, editors, *iTrust*, volume 3477 of *Lecture Notes in Computer Science*, pages 45–60. Springer, 2005.
- [13] Sviatoslav Braynov and Murtuza Jadiwala. Representation and Analysis of Coordinated Attacks. In *FMSE '03: Proceedings of the 2003 ACM workshop on Formal methods in security engineering*, pages 43–51, New York, NY, USA, 2003. ACM Press.
- [14] Axel van Lamsweerde. Elaborating Security Requirements by Construction of Intentional Anti-Models. In *ICSE '04: Proceedings of the 26th International Conference on Software Engineering*, pages 148–157, Washington, DC, USA, 2004. IEEE Computer Society.
- [15] Oleg Sheyner, Joshua Haines, Somesh Jha, Richard Lippmann, and Jeannette M. Wing. Automated Generation and Analysis of Attack Graphs. In *SP '02: Proceedings of the 2002 IEEE Symposium on Security and Privacy*, page 273, Washington, DC, USA, 2002. IEEE Computer Society.
- [16] Alexandros Xenakis and Ann Macintosh. Procedural Security Analysis of Electronic Voting. In *ICEC '04: Proceedings of the 6th international conference on Electronic commerce*, pages 541–546, New York, NY, USA, 2004. ACM Press.
- [17] Alexandros Xenakis and Ann Macintosh. Procedural Security and Social Acceptance in E-Voting. In *HICSS '05: Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 5*, page 118.1, Washington, DC, USA, 2005. IEEE Computer Society.
- [18] Alexandros Xenakis and Ann Macintosh. Using Business Process Re-engineering (BPR) for the Effective Administration of Electronic Voting. *The Electronic Journal of e-Government*, 3(2), 2005.